

Single Particle Algorithms for Continuous Optimization

Giovanni Iacca*, Fabio Caraffini ^{† ‡}, Ferrante Neri ^{† ‡}, Ernesto Mininno[‡]

*INCAS³ - Innovation Centre for Advanced Sensors and Sensor Systems
P.O. Box 797, 9400 AT Assen, The Netherlands
Email:giovanniiacca@incas3.eu

[†] Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University,
The Gateway, Leicester LE1 9BH, United Kingdom
Email: fabio.caraffini@email.dmu.ac.uk and fneri@dmu.ac.uk

[‡] Department of Mathematical Information Technology,
P.O. Box 35 (Agora), 40014 University of Jyväskylä, Finland
Email: fabio.caraffini@jyu.fi, ferrante.neri@jyu.fi, and ernesto.mininno@jyu.fi

Abstract—This paper introduces two lightweight variants of ISPO, a Single Particle Optimization algorithm recently proposed in the literature. The goal of this work is to improve upon the performance of the original ISPO, still bearing in mind its admirable algorithmic simplicity. The first variant, namely ISPO-restart, combines in a memetic fashion the logics of ISPO with a partial restart mechanism similar to the binomial crossover typically used in Differential Evolution. The second variant, named VISPO, builds on top of the restart process a very simple learning stage which tries to adapt the algorithm behaviour to the (non)-separability of the problem. Numerical results obtained on three complete optimization benchmarks show that not only the two algorithms are able to improve, incrementally, upon the performance of ISPO, but also they show respectable performance in comparison with modern complex state-of-the-art methods, especially when the problem dimensionality increases.

I. INTRODUCTION

The increasing complexity of optimization problems led, in recent years, to an increase in complexity of optimization methods. Challenges in optimization can be due to the large scale nature of some applications, highly multi-modal landscape, ill-conditioning, non-separability of the objective function (especially if associated to a high dimensionality), etc. In order to tackle these issues, modern algorithms often make use of sophisticated techniques such as multiple local search activations [1], machine learning structures [2], coordinated/simultaneous action of multiple paradigms, see [3], [4], and [5], sampling from distributions to coordinate different search logics [6], and an archive [7].

As highlighted in [8] and [9], the complexity in algorithmic design often pays off with an increase in the memory requirements and computational overhead. In addition, complex algorithms are hard to control and adapt to new optimization scenarios since the effect of each operator is likely to be unclear. Most importantly, the complexity of modern algorithms is not always rewarded with a high performance. Conversely, the two memetic structures proposed in [8] and [9] have shown

how a simplistic approach can easily be as good as (if not better than) modern algorithms composed of multiple parts and making use of sophisticated coordination schemes.

In this light, the authors propose, while performing the algorithmic design, a bottom-up strategy that ensures the understanding of the role and action of each operator, see [8] and [9]. If we consider Memetic Computing (MC) approaches as structured collection of heterogeneous (and diverse) operators, the design of an optimization algorithm coincides with the design of a MC approach. The present paper, in a memetic fashion, and making use of the law of simplicity in MC stated in [8], revisits an algorithm recently proposed in [10], namely Intelligence Single Particle Optimization (ISPO), and enhances upon its performance without including any major complexity to the original scheme.

The ISPO algorithm is characterized by a very simple structure that randomly perturbs each variable at once. Although this search logic appears to be effective only for separable problems, ISPO appears to be able to display a respectable performance also in non-separable cases, see [10] and [11]. In this paper, we propose a first enhancement of the ISPO scheme by including a simple restart logic, then we propose an ISPO variant that adapts (in an extremely simplistic way), the number of perturbations at each step of the optimization process.

The remainder of this paper is organized in the following way. Section II describes the ISPO structure, its variant employing a restart mechanism and the proposed variant employing restart and learning mechanism. The latter ISPO variant is here referred to as Very Intelligent Single Particle Optimization (VISPO). Section III displays, on a set of diverse problems, the performance of the proposed approach with respect to that of the original ISPO and that of modern popular population-based algorithms. Finally, Section IV gives the conclusion of this study.

II. ALGORITHMIC DESIGN

A. Original ISPO

The original ISPO, proposed in [10], can be considered as a “degenerate” PSO algorithm which explores the search space using a single n -dimensional particle $\mathbf{x}$, rather than an actual swarm. At the beginning of the optimization, the particle is initialized with a random point in the search space. At each step of the algorithm, the following operations are then performed. For each i -th variable of $\mathbf{x}$, a learning factor L is initialized to zero. Then the i -th variable is perturbed H times: for each t -th perturbation ($t = 1, 2, \dots, H$), a velocity factor v , computed as:

$$v = A/t^P \cdot \text{rand}(-0.5, 0.5) + B \cdot L^t \quad (1)$$

is added to the previous value of the i -th variable, as in a “standard” PSO ($x_i = x_i + v$). Parameters A , P , and B are the acceleration, acceleration power factor, and learning coefficient, respectively. We indicate with $\text{rand}(x, y)$ a random number drawn from a uniform distribution in $[x, y]$. After the i -th variable is perturbed, the fitness of the new particle is calculated and compared with the fitness prior to the perturbation. If an improvement has been achieved (or the perturbed solution has the same fitness of the original particle), the learning factor is updated as $L = v$, otherwise it is reduced by means of a shrinking factor S_f : $L = L/S_f$. If L becomes smaller than a precision value ε , then it is reinitialized to zero. For the sake of clarity, the pseudo-code of ISPO is shown in Fig. 1, where the stop criterion is intentionally left unspecified (it can be e.g. a condition on the number of fitness evaluations, a maximum computational time, etc.).

```

1: initialize particle  $\mathbf{x}$ 
2: initialize  $A, P, B, S_f, \varepsilon, H$ 
3: while stop criterion is not met do
4:   for  $i = 1 \rightarrow n$  do
5:      $L = 0$ 
6:     for  $t = 1 \rightarrow H$  do
7:       save old position  $x_i^{\text{bak}} = x_i$ 
8:       calculate  $v$  according to eq. (1)
9:       update position  $x_i = x_i + v$ 
10:      calculate fitness  $f(\mathbf{x})$ 
11:      if fitness improvement then
12:        use velocity as learning factor  $L = v$ 
13:      else
14:        if  $L \neq 0$  then
15:          adjust learning factor  $L = L/S_f$ 
16:        end if
17:        if  $L < \varepsilon$  then
18:           $L = 0$ 
19:        end if
20:        restore old position  $x_i = x_i^{\text{bak}}$ 
21:      end if
22:    end for
23:  end for
24: end while

```

Fig. 1. ISPO pseudo-code

Despite its simplicity, ISPO has proven to be competitive on numerous benchmark functions characterized by different

properties. On the other hand, the main limitation of ISPO is its intrinsic exploitation pressure which is not properly counterbalanced by exploration. Additionally, according to [10] the parameters A and P are heavily problem-dependent and finding a proper setting is far from being a trivial task.

In order to overcome these limitations, we propose here two minor modifications of ISPO, i.e. a simplification of the update eq. (1), and the introduction of two simple mechanisms, namely restart an learning, in the original structure.

B. First variant: ISPO-restart

The first variant we propose, namely ISPO-restart, introduces two novelties in the original ISPO structure. The first one relates to the acceleration A used in eq. (1), which as said is problem-dependent. In ISPO-restart, rather than using the same, fixed value of A for all the decision variables, we use a different value of acceleration for each i -th variable. Each component A_i of the acceleration is initialized to the width of the search space along that dimension ($ub_i - lb_i$, where lb_i and ub_i are, respectively, the lower and upper bounds for the i -th variable). This way each dimension is exploited differently, proportionally to the width of search space along it.

The second, more important modification introduced in this variant is the restart mechanism. As shown in Fig. 2, this mechanism adopts a logics similar to the binomial crossover used in Differential Evolution [12]. More specifically, whenever the restart is activated (see below), a dynamically adapting crossover rate CR is computed based on the current and maximum number of fitness evaluations, respectively n_{eval} and N_{eval} . The crossover rate is updated in such a way that it takes its maximum value at the beginning of the optimization, and decreases during the next stages. As it will be clear in the following, this mechanism promotes a stronger exploration at first, and favors gradually the exploitation later on.

Once CR has been computed, for each i -th variable of the current best particle a uniform random number r in $[0, 1]$ is generated and compared with it. If $r < CR$, the i -th variable is re-sampled within its bounds; otherwise it will retain its original value. The modified particle $\mathbf{x}^*$ is then evaluated, and, in case of fitness improvement, it is copied into the current best particle. The latter is finally passed to the next step of the algorithm, and the process continues until a stop condition is met. The pseudo code of ISPO-restart is shown in Fig. 3. As a final remark, it should be noted that the restart is activated based on a predetermined threshold on the fitness improvement: if a “macro-iteration” of ISPO (see Fig. 3) produces a *relative* fitness improvement smaller than the threshold, a partial re-sampling of the particle is attempted.

The idea behind ISPO-restart is simple: compared to ISPO, it introduces a restart mechanism which tends to re-sample the current particle when ISPO is not able to improve enough upon it. This re-sampling mechanism guarantees a better trade-off between exploration and exploitation in two ways: on one hand, it retains parts of the current particle (rather than sampling a completely new solution), thus preserving part of the previous achievements; on the other, the adaptive crossover rate adjusts the amount of variables which are inherited from the original particle.

```

1: save current best particle  $\mathbf{x}$ 
2: adaptive crossover rate  $CR = [(N_{eval} - n_{eval})/N_{eval}]^2$ 
3: for  $i = 1 \rightarrow n$  do
4:   if  $rand(0, 1) < CR$  then
5:      $x_i^* = rand(lb_i, ub_i)$ 
6:   else
7:      $x_i^* = x_i$ 
8:   end if
9: end for
10: if  $f(\mathbf{x}^*) < f(\mathbf{x})$  then
11:    $\mathbf{x} = \mathbf{x}^*$ 
12: end if

```

Fig. 2. Restart pseudo-code

```

1: initialize particle  $\mathbf{x}$ 
2: initialize  $P, B, S_f, \varepsilon, H, threshold$ 
3: for  $i = 1 \rightarrow n$  do
4:   initialize  $A_i = (ub_i - lb_i)$ 
5: end for
6: while stop criterion is not met do
7:   // original ISPO
8:   save current best fitness  $f_1$ 
9:   execute a step of ISPO (Fig. 1, lines 4-23)
10:  save current best fitness  $f_2$ 
11:  // restart
12:  if  $(f_1 - f_2)/f_1 \leq threshold$  then
13:    perform restart as in Fig. 2
14:  end if
15: end while

```

Fig. 3. ISPO-restart pseudo-code

C. Second variant: VISPO

The second ISPO variant we propose here, namely VISPO (standing for Very Intelligent Single Particle Optimization), represents a further development of the ISPO concept. In this variant, on top of the re-sampling mechanism introduced in ISPO-restart, we build a minimalistic learning process which tries to adapt the algorithmic behaviour to the level of separability of the optimization problem at hand. Additionally, the update equation (1) used in ISPO and ISPO-restart is further simplified in order to reduce the number of parameters required.

With reference to Fig. 4, which shows the pseudo-code of VISPO, the algorithm consists of the following. Similar to ISPO-restart, a parameter A_i is initialized for each variable to the width of the search space along that dimension. Also, the same restart mechanism introduced in ISPO-restart is used to re-sample the particle when a macro-iteration of the algorithm does not produce any improvement. In this case the trigger condition for the restart is more restrictive, though, as the fitness threshold is set to zero.

During each macro-iteration, as seen before each i -th variable is perturbed H times, where H is a parameter of the algorithm. However, in VISPO the initial velocity is calculated *before* applying the H perturbations. Furthermore, the velocity update rule is simpler than that one used in the two previous algorithms:

$$v = A_i \cdot rand(-0.5, 0.5). \quad (2)$$

The velocity v is then added to the i -th variable x_i and the perturbed solution is evaluated. In case the particle prior to the perturbation was better than the perturbed one, the velocity v is halved. This way incrementally smaller perturbations are attempted. On the other hand, if the perturbation produces a fitness improvement, the same velocity is used for the next perturbation and a counter s (“success”), which is initialized to zero at the beginning of the optimization, is incremented.

The counter s is used, after LP (learning period) “macro-iterations” (consisting of $n \times H$ trials), to learn, empirically, the level of separability of the fitness function. More specifically, if the quantity $s/(n \cdot LP)$ is smaller than a given threshold, the number of perturbations H which will be applied in the next stages of the algorithm is set to one, otherwise the initial value of H is retained.

This mechanism can be explained as follows. If the percentage of successful perturbations, averaged over the problem dimension n and the number of learning periods LP , is smaller than a threshold, then the problem can be considered, with some degree of approximation, non-separable. In this case it makes sense to attempt, in later stages of the algorithm, a single perturbation for each dimension. On the other hand, if this percentage is bigger than the given threshold (meaning that during the learning periods an averagely high number of successful perturbations was obtained per each variable) the level of separability of the problem can be considered higher. Thus it is worthwhile to keep on trying multiple perturbations per dimension, aiming at exploiting the separability of the fitness function.

The mechanism described above is as said rather minimalistic, and based on empirical rules. It represents a simple attempt to implement a learning rule which is able to determine the level of separability of a given fitness function. More complex and rigorous methods can be envisioned and are currently under investigation, with the final goal of creating an intelligent agent capable of automatically design an algorithm based on the characteristics of the optimization problem at hand. Still, despite the simplicity of the mechanism implemented in VISPO, numerical results, discussed in the next section, show that it provides a large performance improvement with respect to ISPO and ISPO-restart.

III. NUMERICAL RESULTS

In order to assess the scalability of the the two proposed variants of ISPO, ISPO-restart and VISPO have been tested, together with the original ISPO implementation, on two complete benchmarks used in Computational Intelligence Optimization, namely:

- CEC 2005 [13] (25 test functions, with $n = 30$);
- BBOB 2010 [14] (24 test functions, with $n = 100$);
- CEC 2010 [15] (20 test functions, with $n = 1000$).

Thus 69 fitness functions have been tested in total, with a dimensionality ranging from 30 to 1000. The three ISPO-based algorithms have also been compared with the following algorithms:

```

1: initialize particle  $\mathbf{x}$ 
2: initialize  $LP, H, threshold$ 
3: for  $i = 1 \rightarrow n$  do
4:   initialize  $A_i = (ub_i - lb_i)$ 
5: end for
6: initialize  $s = 0$ 
7: initialize  $k = 0$ 
8: while stop criterion is not met do
9:   // modified ISPO
10:  save current best fitness  $f_1$ 
11:  for  $i = 1 \rightarrow n$  do
12:    calculate  $v$  according to eq. (2)
13:    for  $t = 1 \rightarrow H$  do
14:      save old position  $x_i^{bak} = x_i$ 
15:      update position  $x_i = x_i + v$ 
16:      calculate fitness  $f(\mathbf{x})$ 
17:      if fitness improvement then
18:         $s = s + 1$ 
19:      else
20:        adjust learning factor  $v = v/2$ 
21:        restore old position  $x_i = x_i^{bak}$ 
22:      end if
23:    end for
24:  end for
25:  save current best fitness  $f_2$ 
26:  // learning complete
27:  if  $k == LP$  then
28:    if  $s/(n \cdot LP) < threshold$  then
29:       $H = 1$ 
30:    end if
31:  end if
32:  // restart
33:  if  $(f_1 - f_2) == 0$  then
34:    perform restart as in Fig. 2
35:  end if
36:   $k = k + 1$ 
37: end while

```

Fig. 4. VISPO pseudo-code

- Comprehensive Learning Particle Swarm Optimizer (CLPSO) proposed in [16] with population size equal to 60 individuals;
- Adaptive Differential Evolution (JADE) proposed in [7] with population size equal to 60 individuals, group size factor $p = 0.05$ and parameters adaptation rate factor $c = 0.1$;

which belong to the state-of-the-art continuous optimization methods. The parameter setting for both CLPSO and JADE was chosen following the suggestions in the original papers where the two algorithms have been presented.

As for ISPO, it was executed with $A = 1$, $P = 10$, $B = 2$, $S_f = 4$, $\varepsilon = 10^{-5}$, $H = 30$. The same parameter setting was chosen also for ISPO-restart, except that A in this case was initialized with respect to the search space bounds; the restart threshold was set to 0.01. It should be noted that the values of B , S_f , ε and H are those suggested in [10], while for A and P (which as said are problem-specific) we performed a parameter tuning on the selected benchmarks and picked the values showing the best performance trade-off at various

dimensionalities. Finally, VISPO was executed with $int LP = 10$, $H = 30$ and learning threshold 0.65; also in this case we chose the parameters producing the best overall performance on the benchmarks under analysis.

For each test function composing the aforementioned benchmarks, each algorithm has been executed 100 times in order to draw meaningful statistical conclusions. Each run was continued for $5000 \times n$ fitness evaluations, being n the problem dimension. All the experiments were coded in Java using the distributed platform Kimeme [17].

We should also remark that all the algorithms analysed in this study make use of a toroidal handling of the search space bounds: in other words, if x_i exceeds its upper bound ub_i by a quantity ζ ($x_i = ub_i + \zeta$), the replacement $x_i = lb_i + \zeta$ occurs. A similar mechanism is applied when x_i exceeds its lower bound.

Numerical results in this section are organized into two groups. In the first group, the performance of the three ISPO variants are compared, see Table I, II, and III respectively for CEC 2005, BBOB 2010 and CEC 2010 benchmarks. In the second group, the most promising ISPO variant, i.e. the VISPO algorithm, is compared against CLPSO and JADE, see IV, V, and VI respectively for CEC 2005, BBOB 2010 and CEC 2010 benchmarks.

Each table displays, for each test problem, the average (over the 100 runs available) final fitness $\pm$ standard deviation. In order to strengthen the statistical significance of the results, for each test problem the Wilcoxon Rank-Sum test [18] has been also applied, with a confidence level of 0.95. The symbols “=” and “+” (“-”) indicate, respectively, a statistically equivalent performance and a better (worse) performance of VISPO compared with the algorithm in the column label.

The comparison among the ISPO variants shows that VISPO displays on average the best performance throughout the three test-beds at the three different dimensionality values. VISPO systematically out-performs the original ISPO on all the three benchmarks, particularly in 30 and 100 dimensions. In the 1000-dimensional case, ISPO is slightly better than VISPO only on some unimodal functions (f_1, f_4, f_9, f_{14}), while in the other cases VISPO improves upon it. Similar considerations can be done on the comparison between VISPO and ISPO-restart, which in turns proves slightly superior to ISPO especially in 30 and 100 dimensions. A particularly interesting case is shown in Fig. 5, where VISPO converges toward the optimal solution within the first 10^5 functional calls, while ISPO and ISPO-restart are unable to solve the problem.

As for the comparison with CLPSO and JADE, we can notice that, on the CEC 2005 benchmark in 30 dimensions, VISPO displays a respectable performance both with respect to CLPSO and SADE. To highlight the importance of this result, it should be remembered that CLPSO and SADE, which are considered among the state-of-the-art in optimization, were specifically tuned for this benchmark. Especially the comparison with CLPSO, which is a rather complex PSO algorithm, shows that the simple structure of VISPO performs better in 15 cases out of 25. On the other hand, the comparison with JADE shows a substantially equivalent performance (12 “+” and 13 “-”). This situation is reversed when the problem dimensionality increases, especially in 100 dimensions, where

TABLE VII. HOLM TEST ON THE FITNESS, REFERENCE ALGORITHM = VISPO (RANK = 3.42E+00)

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
VISPO2	CLPSO	3.01e+00	-1.85e+00	3.24e-02	5.00e-02	Rejected

VISPO performs better than JADE, particularly on separable and unimodal functions, and shows the same performance as CLPSO. A similar trend can also be observed in 1000 dimensions, where once again the simplicity of VISPO is able to deal with the curse of dimensionality at least as well as the complex structures of CLPSO and JADE. An example of this trend is displayed in Fig. 6, which compares the fitness trend of the three algorithms under investigation on the sphere problem in 1000 dimensions.

In addition to the results presented above, the ranking among all the algorithms considered in this article has been performed by means of the Holm-Bonferroni procedure, see [19] and [20], for the 5 algorithms under study and the 69 problems under consideration. The Holm-Bonferroni procedure consists of the following. Considering the results in the tables above, the 5 algorithms under analysis have been ranked on the basis of their average performance calculated over the 69 test problems. More specifically, a score R_i for $i = 1, \dots, N_A$ (where N_A is the number of algorithms under analysis, $N_A = 5$ in our case) has been assigned. The score has been assigned in the following way: for each problem, a score of 5 is assigned to the algorithm displaying the best performance, 4 is assigned to the second best, 3 to the third and so on. The algorithm displaying the worst performance scores 1. For each algorithm, the scores obtained on each problem are summed up averaged over the amount of test problems (69 in our case). On the basis of these scores the algorithms are sorted (ranked). With the calculated R_i values, VISPO has been taken as a reference algorithm. Indicating with R_0 the rank of VISPO, and with R_j for $j = 1, \dots, N_A - 1$ the rank of one of the remaining eleven algorithms, the values z_j have been calculated as $z_j = (R_j - R_0) / \sqrt{(N_A(N_A + 1)) / (6N_{TP})}$ where N_{TP} is the number of test problems in consideration ($N_{TP} = 69$ in our case). By means of the z_j values, the corresponding cumulative normal distribution values p_j have been calculated. These p_j values have then been compared with the corresponding δ/j where δ is the level of confidence, set to 0.05 in our case. Table VII displays the ranks, z_j values, p_j values, and corresponding δ/j obtained in this way. The rank of VISPO is shown in parenthesis. Moreover, it is indicated whether the null-hypothesis (that the two algorithms have indistinguishable performances) is “Rejected”, i.e. VISPO statistically outperforms the algorithm under consideration, or “Accepted” if the distribution of values can be considered the same (there is no out-performance).

The Holm-Bonferroni procedure in Table VII shows that VISPO displays the best performance, thus outperforming the other ISPO variants as well as CLPSO and JADE. This allows us to conclude that VISPO is a valuable algorithm that can be used in various contexts. The importance of the proposed algorithm is not only due to its simplicity but also due to its modest overhead and memory requirements. This features makes VISPO an appealing scheme for problems characterized by a modest hardware, see [21], [22], and [23].

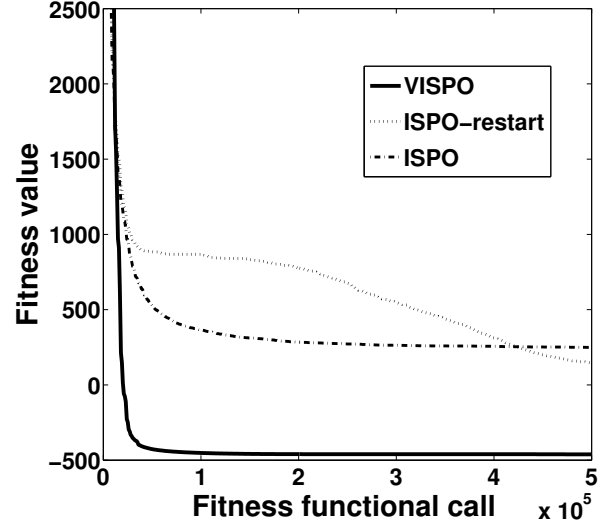


Fig. 5. Average fitness trend of ISPO and its variants for f_4 from BBOB 2010 [14] in 100 dimensions.

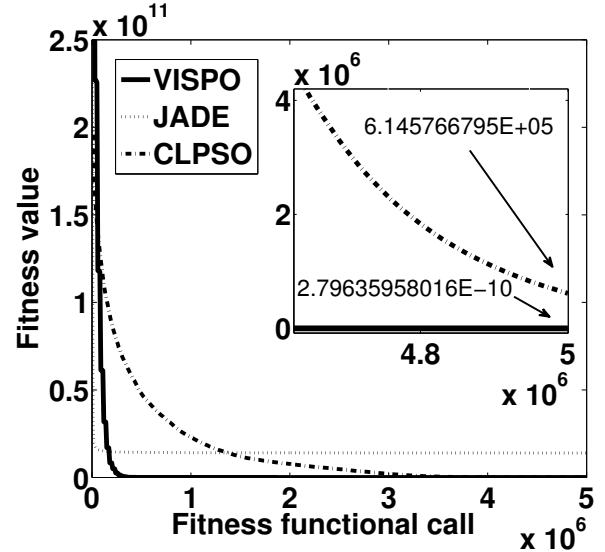


Fig. 6. Average fitness trend of VISPO against population-based algorithms for f_1 from CEC 2010 [15] in 1000 dimensions.

IV. CONCLUSION

In this paper we presented two simple variants of ISPO, a Simple Intelligence Single Particle Optimization algorithm known from the literature and especially suitable for hardware-limited applications. These two variants introduce, with an incremental yet limited increase of algorithmic complexity, a restart and a learning mechanism. The simple structure of ISPO is thus slightly altered in the attempt of improving upon its original performance. The proposed methods, namely ISPO-restart and VISPO, both proved superior with respect to the original ISPO and showed a respectable performance compared with popular state-of-the-art optimization algorithms such as CLPSO and JADE. In particular the VISPO structure, which employs a minimalistic learning strategy and a simple re-sampling mechanism based on the binomial crossover used

TABLE I. AVERAGE FITNESS $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = VISPO) FOR VISPO AGAINST ISPO AND ISPO-RESTART ON CEC2005[13] IN 30 DIMENSIONS.

	VISPO	ISPO		ISPO-restart	
f_1	$-4.50e+02 \pm 6.31e-02$	$-4.50e+02 \pm 6.12e-14$	=	$-4.50e+02 \pm 1.37e-13$	=
f_2	$1.18e+03 \pm 2.05e+03$	$-4.25e+02 \pm 1.28e+01$	-	$-3.26e+02 \pm 3.08e+01$	-
f_3	$7.85e+06 \pm 7.90e+06$	$2.47e+06 \pm 1.13e+06$	-	$3.16e+06 \pm 1.86e+06$	-
f_4	$9.63e+03 \pm 3.30e+03$	$2.79e+06 \pm 2.88e+06$	+	$9.62e+04 \pm 2.31e+04$	+
f_5	$1.44e+04 \pm 4.79e+03$	$2.44e+04 \pm 6.53e+03$	+	$7.01e+03 \pm 1.85e+03$	-
f_6	$1.29e+03 \pm 2.59e+03$	$1.35e+03 \pm 1.93e+03$	=	$2.56e+03 \pm 4.00e+03$	+
f_7	$4.50e+10 \pm 1.85e+11$	$-1.80e+02 \pm 1.25e-02$	-	$1.15e+07 \pm 9.69e+07$	-
f_8	$-1.19e+02 \pm 1.04e-01$	$-1.20e+02 \pm 6.52e-02$	-	$-1.20e+02 \pm 7.61e-02$	-
f_9	$-1.14e+02 \pm 6.34e+00$	$-9.12e+01 \pm 1.24e+01$	+	$-1.16e+02 \pm 2.28e+00$	-
f_{10}	$2.92e+02 \pm 2.76e+01$	$3.07e+02 \pm 3.09e+01$	+	$2.62e+02 \pm 2.63e+01$	-
f_{11}	$1.24e+02 \pm 2.92e+00$	$1.29e+02 \pm 4.12e+00$	+	$1.24e+02 \pm 2.57e+00$	=
f_{12}	$6.22e+04 \pm 6.29e+04$	$1.28e+05 \pm 1.38e+05$	+	$4.23e+04 \pm 3.51e+04$	-
f_{13}	$-1.27e+02 \pm 4.85e-01$	$-1.27e+02 \pm 5.72e-01$	+	$-1.26e+02 \pm 7.72e-01$	+
f_{14}	$-2.87e+02 \pm 3.48e-01$	$-2.85e+02 \pm 2.14e-01$	+	$-2.87e+02 \pm 3.81e-01$	+
f_{15}	$1.44e+03 \pm 2.82e+00$	$1.45e+03 \pm 5.22e+00$	+	$1.45e+03 \pm 1.78e+00$	+
f_{16}	$1.60e+03 \pm 1.80e+01$	$1.62e+03 \pm 3.60e+01$	+	$1.56e+03 \pm 9.94e+00$	-
f_{17}	$1.59e+03 \pm 1.31e+01$	$1.74e+03 \pm 3.52e+01$	+	$1.66e+03 \pm 1.85e+01$	+
f_{18}	$9.10e+02 \pm 1.14e-10$	$9.10e+02 \pm 1.76e-10$	+	$9.10e+02 \pm 4.77e-05$	+
f_{19}	$9.10e+02 \pm 9.02e-11$	$9.10e+02 \pm 9.76e-11$	+	$9.10e+02 \pm 4.60e-05$	+
f_{20}	$9.10e+02 \pm 1.52e-10$	$9.10e+02 \pm 1.42e-10$	+	$9.10e+02 \pm 5.57e-05$	+
f_{21}	$1.73e+03 \pm 1.25e+01$	$1.75e+03 \pm 2.04e+01$	+	$1.71e+03 \pm 8.97e+00$	-
f_{22}	$2.64e+03 \pm 8.82e+01$	$3.02e+03 \pm 1.79e+02$	+	$2.54e+03 \pm 6.71e+01$	-
f_{23}	$1.70e+03 \pm 6.58e+00$	$1.75e+03 \pm 2.10e+01$	+	$1.71e+03 \pm 7.46e+00$	+
f_{24}	$1.69e+03 \pm 1.37e+01$	$1.74e+03 \pm 1.31e+01$	+	$1.71e+03 \pm 1.03e+01$	+
f_{25}	$1.80e+03 \pm 3.89e+02$	$3.03e+04 \pm 9.29e+04$	+	$1.71e+03 \pm 2.18e+02$	=

TABLE II. AVERAGE FITNESS $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = VISPO) FOR VISPO AGAINST ISPO AND ISPO-RESTART ON BBOB2010[14] IN 100 DIMENSIONS.

	VISPO	ISPO		ISPO-restart	
f_1	$7.95e+01 \pm 3.98e-04$	$7.95e+01 \pm 1.34e-13$	=	$7.95e+01 \pm 4.63e-14$	=
f_2	$-2.10e+02 \pm 6.25e-13$	$-2.10e+02 \pm 2.30e-14$	=	$-2.10e+02 \pm 5.33e-14$	=
f_3	$-4.62e+02 \pm 2.54e-01$	$6.38e+01 \pm 1.12e+02$	+	$-2.26e+00 \pm 9.59e+01$	+
f_4	$-4.61e+02 \pm 8.19e-01$	$2.49e+02 \pm 1.60e+02$	+	$1.46e+02 \pm 1.22e+02$	+
f_5	$-9.21e+00 \pm 0.00e+00$	$-9.21e+00 \pm 0.00e+00$	=	$-9.21e+00 \pm 0.00e+00$	=
f_6	$1.40e+02 \pm 4.95e+01$	$2.58e+02 \pm 1.02e+02$	+	$3.45e+02 \pm 1.37e+02$	+
f_7	$4.80e+02 \pm 8.21e+01$	$2.03e+03 \pm 5.20e+02$	+	$1.92e+03 \pm 3.84e+02$	+
f_8	$2.02e+02 \pm 5.27e+01$	$2.41e+02 \pm 5.11e+01$	+	$3.57e+02 \pm 5.58e+01$	+
f_9	$2.36e+02 \pm 3.63e+01$	$2.24e+02 \pm 2.30e+01$	-	$2.48e+02 \pm 7.25e+01$	+
f_{10}	$3.66e+04 \pm 8.76e+03$	$3.13e+04 \pm 8.42e+03$	-	$5.32e+04 \pm 1.64e+04$	+
f_{11}	$6.16e+02 \pm 4.08e+02$	$1.20e+03 \pm 1.34e+02$	+	$7.89e+02 \pm 6.89e+01$	+
f_{12}	$-6.11e+02 \pm 1.60e+01$	$-6.08e+02 \pm 2.07e+01$	=	$-5.69e+02 \pm 3.62e+02$	=
f_{13}	$3.29e+01 \pm 4.88e+00$	$3.31e+01 \pm 3.79e+00$	=	$3.38e+01 \pm 5.72e+00$	=
f_{14}	$-5.23e+01 \pm 2.85e-04$	$-5.23e+01 \pm 5.54e-04$	+	$-5.23e+01 \pm 2.72e-03$	+
f_{15}	$2.43e+03 \pm 3.42e+02$	$8.24e+03 \pm 1.16e+03$	+	$4.09e+03 \pm 3.94e+02$	+
f_{16}	$1.10e+02 \pm 7.29e+00$	$1.20e+02 \pm 7.73e+00$	+	$1.05e+02 \pm 3.64e+00$	-
f_{17}	$-8.18e+00 \pm 1.46e+00$	$2.56e+01 \pm 2.46e+01$	+	$-5.71e-01 \pm 2.85e+00$	+
f_{18}	$1.51e+01 \pm 5.02e+00$	$1.72e+02 \pm 1.43e+02$	+	$4.17e+01 \pm 1.07e+01$	+
f_{19}	$-8.42e+01 \pm 1.26e+01$	$1.54e+01 \pm 2.22e+01$	+	$-6.21e+01 \pm 9.39e+00$	+
f_{20}	$-5.46e+02 \pm 7.07e-02$	$-5.45e+02 \pm 8.56e-02$	+	$-5.45e+02 \pm 1.18e-01$	+
f_{21}	$5.24e+01 \pm 1.38e+01$	$5.28e+01 \pm 1.40e+01$	=	$4.63e+01 \pm 4.63e+00$	-
f_{22}	$-9.80e+02 \pm 1.58e+01$	$-9.84e+02 \pm 1.18e+01$	-	$-9.95e+02 \pm 5.70e+00$	-
f_{23}	$9.77e+00 \pm 6.17e-01$	$1.06e+01 \pm 7.08e-01$	+	$9.95e+00 \pm 3.72e-01$	+
f_{24}	$1.42e+03 \pm 2.04e+02$	$3.71e+03 \pm 3.63e+02$	+	$2.74e+03 \pm 2.33e+02$	+

in Differential Evolution, showed a generally superior performance on a great variety of test problems combining the CEC 2005, BBOB 2010 and CEC 2010 benchmarks, tested at different dimensionalities.

Further studies are needed to develop the ideas introduced in this work: on one hand, different re-sampling mechanisms will be investigated; on the other, an effort will be put to improve the learning strategy described here and develop a mechanism which is able to estimate various features of the landscape, thus adapting the algorithmic structure to them.

ACKNOWLEDGMENT

INCAS³ is co-funded by the Province of Drenthe, the Municipality of Assen, the European Fund for Regional De-

velopment and the Ministry of Economic Affairs, Peaks in the Delta. This research is supported by the Academy of Finland, Akatemiututkija 130600, "Algorithmic design issues in Memetic Computing". The numerical experiments have been carried out on the computer network of the De Montfort University by means of the software for distributed optimization Kimeme [17]. We thank Dr. Lorenzo Picinali, Dr. Nathan Jeffery and David Tunnicliffe for the technical support.

REFERENCES

- [1] D. Molina, M. Lozano, C. Garcia-Martinez, and F. Herrera, "Memetic Algorithms for Continuous Optimization Based on Local Search Chains," *Evolutionary Computation*, vol. 18, no. 1, pp. 27–63, 2010.
- [2] S. D. Handoko, C. K. Kwok, and Y. S. Ong, "Feasibility Structure Modeling: An Effective Chaperon for Constrained Memetic Algorithms,"

TABLE III. AVERAGE FITNESS $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = VISPO) FOR VISPO AGAINST ISPO AND ISPO-RESTART ON CEC2010[15] IN 1000 DIMENSIONS.

	VISPO	ISPO		ISPO-restart	
f_1	$2.80e-10 \pm 5.24e-11$	7.26e-32 $\pm$ 7.23e-31	-	$6.81e-08 \pm 6.77e-07$	+
f_2	6.75e-02 $\pm$ 2.45e-01	$1.39e+04 \pm 4.51e+02$	+	$2.06e+02 \pm 1.37e+01$	+
f_3	2.98e-09 $\pm$ 1.35e-10	$1.99e+01 \pm 1.19e-02$	+	$2.82e+00 \pm 7.42e-02$	+
f_4	$7.43e+12 \pm 2.61e+12$	5.18e+12 $\pm$ 1.72e+12	-	$5.97e+12 \pm 2.34e+12$	-
f_5	7.95e+08 $\pm$ 1.23e+08	$8.50e+08 \pm 1.30e+08$	+	$8.39e+08 \pm 1.50e+08$	+
f_6	$1.98e+07 \pm 6.21e+04$	1.98e+07 $\pm$ 6.80e+04	=	$1.99e+07 \pm 7.52e+04$	=
f_7	2.03e+10 $\pm$ 9.63e+09	$2.53e+10 \pm 1.57e+10$	+	$2.84e+10 \pm 1.68e+10$	+
f_8	$1.11e+09 \pm 2.28e+09$	$4.97e+08 \pm 6.14e+08$	=	3.57e+08 $\pm$ 4.15e+08	=
f_9	$6.51e+07 \pm 6.82e+06$	5.06e+07 $\pm$ 5.40e+06	-	$5.34e+07 \pm 5.52e+06$	-
f_{10}	7.88e+03 $\pm$ 3.42e+02	$1.49e+04 \pm 5.59e+02$	+	$8.08e+03 \pm 3.58e+02$	+
f_{11}	1.99e+02 $\pm$ 2.31e-01	$2.18e+02 \pm 1.84e-01$	+	$2.02e+02 \pm 2.26e-01$	+
f_{12}	1.19e+05 $\pm$ 1.93e+04	$1.57e+05 \pm 2.76e+04$	+	$1.62e+05 \pm 2.90e+04$	+
f_{13}	$4.03e+03 \pm 4.28e+03$	2.06e+03 $\pm$ 1.98e+03	=	$2.08e+03 \pm 2.10e+03$	=
f_{14}	$1.47e+08 \pm 1.06e+07$	1.14e+08 $\pm$ 8.85e+06	-	$1.24e+08 \pm 9.97e+06$	-
f_{15}	1.54e+04 $\pm$ 4.97e+02	$1.55e+04 \pm 4.72e+02$	=	$1.56e+04 \pm 5.19e+02$	+
f_{16}	$3.98e+02 \pm 3.21e-01$	3.97e+02 $\pm$ 2.62e-01	-	$3.97e+02 \pm 3.26e-01$	-
f_{17}	2.45e+05 $\pm$ 3.44e+04	$3.53e+05 \pm 4.34e+04$	+	$3.60e+05 \pm 3.94e+04$	+
f_{18}	$1.85e+04 \pm 1.16e+04$	1.05e+04 $\pm$ 5.02e+03	-	$1.26e+04 \pm 6.59e+03$	-
f_{19}	6.21e+06 $\pm$ 5.17e+05	$3.82e+07 \pm 6.30e+06$	+	$3.82e+07 \pm 7.79e+06$	+
f_{20}	$6.58e+02 \pm 1.75e+02$	$7.61e+02 \pm 2.30e+02$	+	4.96e+02 $\pm$ 1.80e+02	-

TABLE IV. AVERAGE FITNESS $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = VISPO) FOR VISPO AGAINST POPULATION-BASED ALGORITHMS ON CEC2005[13] IN 30 DIMENSIONS.

	VISPO	CLPSO		JADE	
f_1	$-4.50e+02 \pm 6.31e-02$	-4.50e+02 $\pm$ 6.12e-05	-	$-3.51e+02 \pm 1.99e+02$	+
f_2	$1.18e+03 \pm 2.05e+03$	$6.44e+03 \pm 9.22e+02$	+	3.08e+02 $\pm$ 7.32e+02	=
f_3	$7.85e+06 \pm 7.90e+06$	$2.55e+07 \pm 6.25e+06$	+	3.71e+06 $\pm$ 1.77e+06	-
f_4	$9.63e+03 \pm 3.30e+03$	$1.43e+04 \pm 1.70e+03$	+	2.27e+03 $\pm$ 1.68e+03	-
f_5	$1.44e+04 \pm 4.79e+03$	$7.14e+03 \pm 5.68e+02$	-	3.91e+03 $\pm$ 9.18e+02	-
f_6	$1.29e+03 \pm 2.59e+03$	6.08e+02 $\pm$ 4.97e+01	-	$5.07e+06 \pm 1.53e+07$	+
f_7	4.50e+10 $\pm$ 1.85e+11	$1.41e+11 \pm 7.45e+10$	+	$1.69e+13 \pm 1.78e+13$	+
f_8	-1.19e+02 $\pm$ 1.04e-01	$-1.19e+02 \pm 5.05e-02$	-	$-1.19e+02 \pm 5.75e-02$	+
f_9	$-1.14e+02 \pm 6.34e+00$	$-1.15e+02 \pm 8.97e-01$	-	-1.17e+02 $\pm$ 1.22e+00	-
f_{10}	$2.92e+02 \pm 2.76e+01$	$2.73e+02 \pm 1.22e+01$	-	2.02e+02 $\pm$ 2.20e+01	-
f_{11}	$1.24e+02 \pm 2.92e+00$	$1.19e+02 \pm 1.69e+00$	-	1.16e+02 $\pm$ 4.48e+00	-
f_{12}	$6.22e+04 \pm 6.29e+04$	$4.46e+04 \pm 9.39e+03$	=	1.72e+04 $\pm$ 1.54e+04	-
f_{13}	-1.27e+02 $\pm$ 4.85e-01	$-1.20e+02 \pm 8.21e-01$	+	$-1.26e+02 \pm 9.64e-01$	+
f_{14}	-2.87e+02 $\pm$ 3.48e-01	$-2.87e+02 \pm 1.65e-01$	+	$-2.87e+02 \pm 2.02e-01$	+
f_{15}	1.44e+03 $\pm$ 2.82e+00	$1.46e+03 \pm 2.18e+00$	+	$1.45e+03 \pm 3.45e+00$	+
f_{16}	$1.60e+03 \pm 1.80e+01$	$1.61e+03 \pm 4.94e+00$	+	1.56e+03 $\pm$ 6.50e+00	-
f_{17}	1.59e+03 $\pm$ 1.31e+01	$1.67e+03 \pm 6.15e+00$	+	$1.59e+03 \pm 7.53e+00$	+
f_{18}	9.10e+02 $\pm$ 1.14e-10	$9.10e+02 \pm 3.54e-05$	+	$9.10e+02 \pm 2.62e-01$	+
f_{19}	9.10e+02 $\pm$ 9.02e-11	$9.10e+02 \pm 3.06e-05$	+	$9.10e+02 \pm 1.31e-01$	+
f_{20}	9.10e+02 $\pm$ 1.52e-10	$9.10e+02 \pm 3.09e-05$	+	$9.10e+02 \pm 1.40e-01$	+
f_{21}	$1.73e+03 \pm 1.25e+01$	$1.72e+03 \pm 3.69e+00$	-	1.69e+03 $\pm$ 4.32e+00	-
f_{22}	$2.64e+03 \pm 8.82e+01$	$2.55e+03 \pm 1.91e+01$	-	2.29e+03 $\pm$ 3.44e+01	-
f_{23}	$1.70e+03 \pm 6.58e+00$	$1.77e+03 \pm 5.94e+00$	+	1.70e+03 $\pm$ 4.48e+00	-
f_{24}	$1.69e+03 \pm 1.37e+01$	$1.70e+03 \pm 6.21e+00$	+	1.66e+03 $\pm$ 1.40e+01	-
f_{25}	1.80e+03 $\pm$ 3.89e+02	$1.89e+03 \pm 8.46e+01$	+	$1.86e+03 \pm 4.65e+01$	+

- IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 740–758, 2010.
- [3] M. A. Montes de Oca, T. Stutzle, M. Birattari, and M. Dorigo, “Frankenstein’s PSO: A Composite Particle Swarm Optimization Algorithm,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.
- [4] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, “Self-Adaptive Multithreshold Search for Global Optimization in Real-Parameter Spaces,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 243–259, 2009.
- [5] F. Peng, K. Tang, G. Chen, and X. Yao, “Population-Based Algorithm Portfolios for Numerical Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782–800, 2010.
- [6] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.
- [7] J. Zhang and A. C. Sanderson, “JADE: Adaptive Differential Evolution with Optional External Archive,” vol. 13, no. 5, 2009, pp. 945–958.
- [8] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim, “Ockham’s Razor in Memetic Computing: Three Stage Optimal Memetic Exploration,” *Information Sciences*, vol. 188, pp. 17–43, 2012.
- [9] F. Caraffini, F. Neri, G. Iacca, and A. Mol, “Parallel memetic structures,” *Information Sciences*, vol. 227, pp. 60–82, 2013.
- [10] J. Zhou, Z. Ji, and L. Shen, “Simplified Intelligence Single Particle Optimization Based Neural Network for Digit Recognition,” in *Proceedings of the Chinese Conference on Pattern Recognition*, 2008, pp. 1–5, 10311847.
- [11] Z. Zhu, J. Zhou, Z. Ji, and Y.-H. Shi, “Dna sequence compression using adaptive particle swarm optimization-based memetic algorithm,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 643–658, 2011.
- [12] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.
- [13] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, “Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization,” Nanyang Technological University and KanGAL, Singapore and IIT Kanpur, India, Tech. Rep. 2005005, 2005.

TABLE V. AVERAGE FITNESS $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = VISPO) FOR VISPO AGAINST POPULATION-BASED ALGORITHMS ON BBOB2010[14] IN 100 DIMENSIONS.

	VISPO	CLPSO	JADE
f_1	$7.95e+01 \pm 3.98e-04$	$7.95e+01 \pm 4.12e-10$	$8.77e+01 \pm 7.64e+00$
f_2	$-2.10e+02 \pm 6.25e-13$	$-2.10e+02 \pm 9.39e-07$	$5.73e+04 \pm 8.69e+04$
f_3	$-4.62e+02 \pm 2.54e-01$	$-4.62e+02 \pm 4.45e-01$	$-3.11e+02 \pm 4.95e+01$
f_4	$-4.61e+02 \pm 8.19e-01$	$-4.55e+02 \pm 1.50e+00$	$-1.43e+02 \pm 9.83e+01$
f_5	$-9.21e+00 \pm 0.00e+00$	$2.58e+02 \pm 1.17e+01$	$1.24e+02 \pm 5.08e+01$
f_6	$1.40e+02 \pm 4.95e+01$	$4.25e+02 \pm 2.65e+01$	$3.92e+02 \pm 1.18e+02$
f_7	$4.80e+02 \pm 8.21e+01$	$2.25e+02 \pm 1.07e+01$	$3.08e+02 \pm 6.50e+01$
f_8	$2.02e+02 \pm 5.27e+01$	$2.82e+02 \pm 3.53e+01$	$7.24e+03 \pm 6.15e+03$
f_9	$2.36e+02 \pm 3.63e+01$	$2.21e+02 \pm 5.45e-01$	$1.78e+03 \pm 1.33e+03$
f_{10}	$3.66e+04 \pm 8.76e+03$	$5.32e+05 \pm 6.46e+04$	$1.94e+05 \pm 8.87e+04$
f_{11}	$6.16e+02 \pm 4.08e+02$	$3.18e+02 \pm 1.67e+01$	$2.11e+02 \pm 2.76e+01$
f_{12}	$-6.11e+02 \pm 1.60e+01$	$-6.03e+02 \pm 5.15e+00$	$2.22e+07 \pm 2.13e+07$
f_{13}	$3.29e+01 \pm 4.88e+00$	$5.08e+01 \pm 2.08e+00$	$7.69e+02 \pm 2.46e+02$
f_{14}	$-5.23e+01 \pm 2.85e-04$	$-5.23e+01 \pm 2.77e-03$	$-4.65e+01 \pm 3.89e+00$
f_{15}	$2.43e+03 \pm 3.42e+02$	$2.10e+03 \pm 3.63e+01$	$1.59e+03 \pm 8.67e+01$
f_{16}	$1.10e+02 \pm 7.29e+00$	$9.53e+01 \pm 2.04e+00$	$1.01e+02 \pm 3.46e+00$
f_{17}	$-8.18e+00 \pm 1.46e+00$	$-1.21e+01 \pm 2.96e-01$	$-1.45e+01 \pm 5.96e-01$
f_{18}	$1.51e+01 \pm 5.02e+00$	$1.12e+00 \pm 1.09e+00$	$-8.79e+00 \pm 2.11e+00$
f_{19}	$-8.42e+01 \pm 1.26e+01$	$-9.48e+01 \pm 2.29e-01$	$-9.50e+01 \pm 2.26e-01$
f_{20}	$-5.46e+02 \pm 7.07e-02$	$-5.45e+02 \pm 6.24e-02$	$-5.12e+02 \pm 9.92e+01$
f_{21}	$5.24e+01 \pm 1.38e+01$	$4.17e+01 \pm 1.21e+00$	$4.93e+01 \pm 6.25e+00$
f_{22}	$-9.80e+02 \pm 1.58e+01$	$-9.98e+02 \pm 5.95e-01$	$-9.94e+02 \pm 6.66e+00$
f_{23}	$9.77e+00 \pm 6.17e-01$	$1.03e+01 \pm 2.74e-01$	$1.07e+01 \pm 3.78e-01$
f_{24}	$1.42e+03 \pm 2.04e+02$	$1.21e+03 \pm 4.50e+01$	$1.03e+03 \pm 4.44e+01$

TABLE VI. AVERAGE FITNESS $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = VISPO) FOR VISPO AGAINST POPULATION-BASED ALGORITHMS ON CEC2010[15] IN 1000 DIMENSIONS.

	VISPO	CLPSO	JADE
f_1	$2.80e-10 \pm 5.24e-11$	$6.15e+05 \pm 1.42e+05$	$1.40e+10 \pm 6.91e+09$
f_2	$6.75e-02 \pm 2.45e-01$	$1.99e+02 \pm 1.27e+01$	$4.56e+03 \pm 1.04e+03$
f_3	$2.98e-09 \pm 1.35e-10$	$9.04e-01 \pm 1.06e-01$	$1.76e+01 \pm 6.75e-01$
f_4	$7.43e+12 \pm 2.61e+12$	$1.22e+13 \pm 2.82e+12$	$2.62e+12 \pm 1.03e+12$
f_5	$7.95e+08 \pm 1.23e+08$	$1.93e+08 \pm 2.20e+07$	$8.58e+07 \pm 1.77e+07$
f_6	$1.98e+07 \pm 6.21e+04$	$9.51e+03 \pm 2.09e+04$	$3.48e+06 \pm 1.40e+06$
f_7	$2.03e+10 \pm 9.63e+09$	$5.53e+08 \pm 1.30e+08$	$3.37e+09 \pm 3.66e+09$
f_8	$1.11e+09 \pm 2.28e+09$	$7.53e+07 \pm 2.77e+07$	$6.31e+13 \pm 1.80e+14$
f_9	$6.51e+07 \pm 6.82e+06$	$9.54e+08 \pm 5.86e+07$	$1.67e+10 \pm 5.87e+09$
f_{10}	$7.88e+03 \pm 3.42e+02$	$6.36e+03 \pm 1.92e+02$	$7.50e+03 \pm 1.07e+03$
f_{11}	$1.99e+02 \pm 2.31e-01$	$9.71e+01 \pm 7.27e+00$	$1.94e+02 \pm 7.49e+00$
f_{12}	$1.19e+05 \pm 1.93e+04$	$7.77e+05 \pm 3.78e+04$	$2.32e+06 \pm 4.55e+05$
f_{13}	$4.03e+03 \pm 4.28e+03$	$9.09e+03 \pm 2.13e+03$	$8.02e+10 \pm 4.76e+10$
f_{14}	$1.47e+08 \pm 1.06e+07$	$1.45e+09 \pm 9.35e+07$	$1.31e+10 \pm 4.64e+09$
f_{15}	$1.54e+04 \pm 4.97e+02$	$1.25e+04 \pm 3.21e+02$	$8.51e+03 \pm 1.03e+03$
f_{16}	$3.98e+02 \pm 3.21e-01$	$2.42e+02 \pm 1.61e+01$	$3.83e+02 \pm 1.19e+01$
f_{17}	$2.45e+05 \pm 3.44e+04$	$1.80e+06 \pm 6.51e+04$	$2.63e+06 \pm 7.56e+05$
f_{18}	$1.85e+04 \pm 1.16e+04$	$7.08e+04 \pm 1.22e+04$	$4.42e+11 \pm 1.91e+11$
f_{19}	$6.21e+06 \pm 5.17e+05$	$5.82e+06 \pm 2.81e+05$	$3.59e+06 \pm 7.17e+05$
f_{20}	$6.58e+02 \pm 1.75e+02$	$3.65e+04 \pm 7.28e+03$	$5.48e+11 \pm 2.10e+11$

- [14] N. Hansen, A. Auger, S. Finck, R. Ros *et al.*, “Real-Parameter Black-Box Optimization Benchmarking 2010: Noiseless Functions Definitions,” INRIA, Tech. Rep. RR-6829, 2010.
- [15] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, “Benchmark Functions for the CEC’2010 Special Session and Competition on Large-Scale Global Optimization,” University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL): Hefei, Anhui, China, Tech. Rep., 2010.
- [16] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.
- [17] Cyber Dyne Srl Home Page, “Kimeme,” 2012, <http://cyberdynesoft.it/>.
- [18] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
- [19] S. Holm, “A simple sequentially rejective multiple test procedure,” *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.
- [20] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, “A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability,” *Soft Computing*, vol. 13, no. 10, pp. 959–977, 2008.
- [21] E. Mininno, F. Neri, F. Cupertino, and D. Naso, “Compact Differential Evolution,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 32–54, 2011.
- [22] F. Neri and E. Mininno, “Memetic Compact Differential Evolution for Cartesian Robot Control,” *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 54–65, 2010.
- [23] F. Neri, G. Iacca, and E. Mininno, “Disturbed Exploitation compact Differential Evolution for Limited Memory Optimization Problems,” *Information Sciences*, vol. 181, no. 12, pp. 2469–2487, 2011.